ERDC MSRC PET Preprint No. 01-29

An MPI Quasi Time-Accurate Approach for Nearshore Wave Prediction Using the SWAN Code

by

Stephen F. Wornom

13 August 2001

Work funded by the Department of Defense
High Performance Computing Modernization Program
U.S. Army Engineer Research and Development Center
Major Shared Resource Center through

Programming Environment and Training

Supported by Contract Number: DAHC94-96-C0002
Computer Sciences Corporation

PET Preprint

An MPI Quasi Time-Accurate Approach for Nearshore Wave Prediction Using the SWAN Code

Stephen F. Wornom
U.S. Army Engineer Research and Development Center
Major Shared Resource Center
Vicksburg, MS

Abstract

The program SWAN (*S*imulating *WA*ves *N*earshore) is a third-generation wave model used to compute the spectra of random short-crested, wind-generated waves on Eulerian grids. No parallel version of the SWAN code for parallel high-performance computing (HPC) platforms currently exists. Work on a parallel version is in its early stages and is unlikely to be available in the near future. This study introduces a quasi time-accurate method in which the individual integrations can be regarded as disjointed so that coarse-grain parallelism may be exploited. The Message Passing Interface (MPI) system is used to pass the data to the parallel processes. For a test case with 72 hours of data, a speedup of 59 using 72 processes was observed. The quasi time-accurate wall time was reduced from 3.34 hours to 3.31 minutes. The wall time for the time-accurate 72-hour computation was 4.58 hours. The differences between the time-accurate results and the quasi time-accurate results were very small for the test case examined. In fact, error norms showed the quasi time-accurate version to be more accurate than the time-accurate version at three of the five test sites. Thus the MPI quasi time-accurate approach may play an important interim role until an efficient time-accurate version of SWAN becomes available. The efficiency of the MPI quasi time-accurate approach opens the possibility of using refined grids and more frequencies and/or directional angles than is possible with the nonparallel time-accurate version.

Introduction

One of the major challenges in ocean modeling is the accurate prediction of nearshore wave conditions required for environmental impact studies of erosion and sediment transport and also equally important in naval operations. The SWAN code has been developed specifically for nearshore zones where finite-depth effects become important. SWAN has been designed for nonstationary computations but can also be used for stationary computations. For nonstationary computations, an implicit time-accurate algorithm is employed to propogate waves on either a spherical or Cartesian grid. Booij et al. (1999) and Ris et al. (1999) give the details of the SWAN code. The SWAN USER MANUAL – see Holthuijsen et al. (1999) – can be downloaded from the SWAN Web site (http://www.swan.ct.tudelft.nl). SWAN is an open source code.

At present, there is no parallel version of the SWAN code for high-performance computing (HPC) platforms. Work on a parallel version is in its early stages, and an efficient time-accurate version is unlikely to be available in the next 1-2 years. At present, the SWAN code is run on a single processor that can result in very long CPU

times for some simulations and thus is prohibitive for Department of Defense (DoD) applications for which rapid results are desired.

Parallelism can occur at either the coarse-grain level or the fine-grain level. Fine-grain parallelism occurs at the finest level possible in a code, for example, the most inner do loop level or parallelization of a function. Coarse-grain parallelism is the largest part of a code that can be run on different processes. Examples include domain-decomposition methods, multiblock codes, codes using data partitioning, and single programs with multiple data where domains, blocks, partitions, and data sets are executed on different processes. For maximum efficiency, both coarse-grain and fine-grain parallelism should be exploited by combining the Message Passing Interface (MPI) system with OpenMP, MPI with Pthreads, or some other combination.

In order to arrive at a parallel version of the SWAN code, this study introduces a quasi time-accurate approach that allows coarse-grain parallelism to be exploited. In this approach, the time dependency enters the computation through time-varying boundary spectra, wind fields, and current fields; the wave action transport equation is solved as a stationary problem at each time output point. This approach has the advantage that the solution procedure changes from a sequential one to a single program with multiple data, where coarse-grain parallelism can be implemented using the MPI message-passing system.

While the quasi time-accurate approach is new to the SWAN code where the time-accurate solver is used, it is the basis for the STWAVE code, which does not have a time-accurate solver. The developers of the STWAVE code, Smith et al. (2001), solve a steady-state spectral wave equation in the STWAVE code. Smith et al. (2001) explained the basis for solving the stationary wave equation at a specific time output. This approach is applied here to the entire range of time-varying boundary spectra and wind fields. Smith et al. (2001) state that solving a steady-state equation "is appropriate for wave conditions that vary more slowly than the time it takes for waves to transit the computational grid. For wave generation, the steady-state assumption means that the winds have remained sufficiently long for the waves to attain fetch-limited or fully developed conditions." As a rule, Smith et al. (2001) apply STWAVE on meshes not more than five kilometers offshore (the direction from which the waves are arriving). This will be referred to as assumption #1.

General approach

The SWAN code was developed at the Delft University of Technology and is continually being upgraded. Consequently, the basic guideline followed here was to make as few as possible changes to the SWAN code itself. In that way, new releases would require only small modifications to run in parallel using the quasi time-accurate approach. In the MPI quasi time-accurate approach developed in this study, all the MPI calls are contained in the master program, which calls the SWAN program. The use of the MPI system to manage the data and launch the computations is a very simple and attractive choice (a single job with many processes). Rather than use the MPI system, one could write script files to submit the different data as different jobs. For example, instead of submitting one

job using 72 processors, one submits 72 different jobs requiring one processor and devises a logic script that would tell the user when all the different jobs have been completed. The MPI approach seems simpler and more attractive.

Modifications to the SWAN source code

The program module SWAN was changed to a subroutine which is called by a front end interface program that reads the user's INPUT file and creates the files needed by the individual processes. The changes to the SWAN code consisted of adding common statements that pass the process ID and the number of processes to SWAN subroutines reading or writing files. The process ID and number of processes are used to control print statements and to identify the input, output, boundary spectra, and wind files for each process.

Changes to the SWAN input file

Modifications to the SWAN input file are given in Figure 1, which illustrates the simplicity of the MPI implementation. Figure 1 shows the base INPUT file, that is, the input file from which all the other process files are created. The only change for MPI runs is the addition of "**DATE_**" before any file read or written on the different processors. For brevity, only nine **COMPUTE** statements are shown. A more general method is possible that avoids having to add the "**DATE_**" prefix.

```
PROJ 'MPI_SWAN input file' '100'
SET CART
MODE NONSTATIONARY TWODIMENSIONAL
GEN3 JANS
FRIC JON
QUAD
OFF BREAKING
COORD  SPHERICAL
CGRID REGULAR   284.  35.  0.  2.0  2.0  96  96 CIRCLE 24 0.04     1.0 24
 INPGRID BOTTOM  REGULAR   284.  35.  0.    192  192    0.010416667  0.010416667
 READINP BOTTOM 1. './DATE_sub2b_SWN_192x192.dep' 2 FREE
$
 INPGRID WIND REGULAR     277.5           10.0000   0.  135  120    0.5  0.5 &
    NONSTAT 9508290000  1.0  HR  9509130000
 READINP WIND 1.0 SERIES 'DATE_swan_wind.inp' 2 FREE
$
 BOUNDNEST2 WAMNEST 'DATE_LUISsub2_cdate_sp.fbi' UNFORMATTED WKSTAT 284.  35.0
$
 INITIAL DEFAULT
$
$************ OUTPUT REQUESTS ************************
POINTS '44014'   FILE 'DATE_44014_sub2_spherical.pts'
POINTS 'chlv2'   FILE 'DATE_chlv2_sub2_spherical.pts'
POINTS 'dsln7'   FILE 'DATE_dsln7_sub2_spherical.pts'
POINTS 'wr630'   FILE 'DATE_wr630_sub2_spherical.pts'
POINTS 'frf8m'   FILE 'DATE_frf8m_sub2_spherical.pts'
$
SPECOUT '44014' SPEC2D 'DATE_b44014.sp2'                        OUTPUT 9508290000 1. HR
SPECOUT 'chlv2' SPEC2D 'DATE_CHLV2.sp2'                         OUTPUT 9508290000 1. HR
SPECOUT 'dsln7' SPEC2D 'DATE_DSLN7.sp2'                         OUTPUT 9508290000 1. HR
SPECOUT 'wr630' SPEC2D 'DATE_WR630.sp2'                         OUTPUT 9508290000 1. HR
SPECOUT 'frf8m' SPEC2D 'DATE_frf8m.sp2'                         OUTPUT 9508290000 1. HR
$
 TABLE 'COMPGRID' 'DATE_sub2.tec'   XP YP TIME HS DIR PER RTP DEP OUTPUT 9508290000 1. HR
 TABLE '44014'    'DATE_b44014.bou' XP YP TIME HS DIR PER RTP DEP OUTPUT 9508290000 1. HR
 TABLE 'chlv2'    'DATE_CHVL2.bou'  XP YP TIME HS DIR PER RTP DEP OUTPUT 9508290000 1. HR
 TABLE 'dsln7'   'DATE_DSLN7.bou' XP YP TIME HS DIR PER RTP DEP OUTPUT 9508290000 1. HR
 TABLE 'wr630'   'DATE_WR630.bou' XP YP TIME HS DIR PER RTP DEP OUTPUT 9508290000 1. HR
 TABLE 'frf8m'   'DATE_frf8m.bou' XP YP TIME HS DIR PER RTP DEP OUTPUT 9508290000 1. HR
```

```
OUTPUT QUANT PER power=0
$
NUM ACC STAT mxitst=5
COMPUTE STAT 9509080100
COMPUTE STAT 9509080200
COMPUTE STAT 9509080300
COMPUTE STAT 9509080400
COMPUTE STAT 9509080500
COMPUTE STAT 9509080600
COMPUTE STAT 9509080700
COMPUTE STAT 9509080800
COMPUTE STAT 9509080900
STOP
```

Figure 1: SWAN input file used for the parallel computations

Data files

The test case was a simulation of 1995 Hurricane Luis used by Wornom et al. (2000). In that study, the time-accurate version of SWAN was used with the output requested every hour. The wind fields consisted of separate files for each hour; thus no changes were necessary for the MPI quasi time-accurate runs. The boundary spectra came from a WAM computation on a larger regional grid and had been written 12 hours/file. To avoid possible file contention in sense #1, that is, different processes attempting to open the same file, the boundary spectra were rewritten to one file/hour (the output times were hourly). Each process input file contains the names of the files needed for that process; thus file contention in sense #1 is avoided. Each process writes its output file, which appears in the directory from which the program is run.

MASTER program

The major implementation effort was in creating a MASTER program to prepare the input files for each of the processes. The MASTER program calls the SWAN code. All the MPI implementation is contained in the MASTER program. Tasks consist of identifying keywords in the generic SWAN input file such as "**DATE_**", "**wind**", and "**OUTPUT**". Other key words are highlighted in bold letters in Figure 1. The MASTER reads the generic input file twice, the first time to determine the number of data sets to be computed and the second to write the input files for each process (the number of requested output times divided by the number of processes determines the number of data per process).

Load balancing

Since each process has different input/output data, file contention in sense #1 was avoided. The size of the data is the same for each process. However, the boundary spectra and wind fields are different for each process, and the number of iterations to achieve convergence to the wave action transport equation may vary from data set to data set and, therefore, from process to process. The convergences for process 34 and process 1 using 72 processes are shown in Figures 2-3. Note that process 34 required 15 iterations to achieve convergence and process 1, four iterations. Note also that the level of convergence for process 34 is 96.60 percent after four iterations, whereas the convergence requirement is 98 percent, which requires 11 additional iterations. The developers of the SWAN code recognized this type of convergence behavior and included a parameter to limit the maximum number of iterations. Figure 4 compares a

maximum for five iterations computation with unlimited number of iterations; the effect on the solution is negligible.

```
myid =  34  accuracy OK in   1.74 % of wet grid points ( 98.00 % required)
myid =  34  accuracy OK in  72.94 % of wet grid points ( 98.00 % required)
myid =  34  accuracy OK in  93.53 % of wet grid points ( 98.00 % required)
myid =  34  accuracy OK in  96.60 % of wet grid points ( 98.00 % required)
myid =  34  accuracy OK in  93.16 % of wet grid points ( 98.00 % required)
myid =  34  accuracy OK in  94.77 % of wet grid points ( 98.00 % required)
myid =  34  accuracy OK in  96.32 % of wet grid points ( 98.00 % required)
myid =  34  accuracy OK in  96.47 % of wet grid points ( 98.00 % required)
myid =  34  accuracy OK in  96.47 % of wet grid points ( 98.00 % required)
myid =  34  accuracy OK in  96.57 % of wet grid points ( 98.00 % required)
myid =  34  accuracy OK in  97.31 % of wet grid points ( 98.00 % required)
myid =  34  accuracy OK in  97.25 % of wet grid points ( 98.00 % required)
myid =  34  accuracy OK in  97.85 % of wet grid points ( 98.00 % required)
myid =  34  accuracy OK in  97.73 % of wet grid points ( 98.00 % required)
myid =  34  accuracy OK in  98.02 % of wet grid points ( 98.00 % required)
```

Figure 2: Process 34/72: max iterations = unlimited

```
myid =   1  accuracy OK in   0.57 % of wet grid points ( 98.00 % required)
myid =   1  accuracy OK in  12.22 % of wet grid points ( 98.00 % required)
myid =   1  accuracy OK in  69.10 % of wet grid points ( 98.00 % required)
myid =   1  accuracy OK in  98.08 % of wet grid points ( 98.00 % required)
```

Figure 3: Process 1/72: max iteration = unlimited

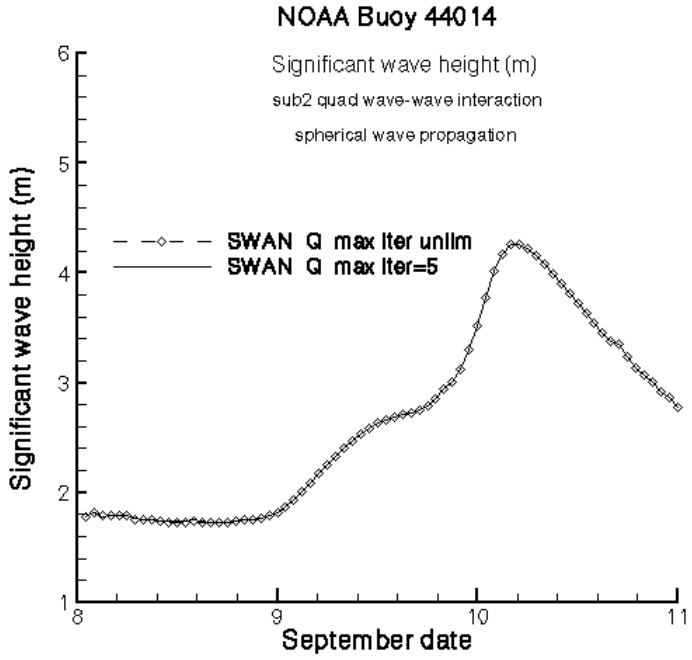

Figure 4: The effect of the number of iterations on the results

MPI speedup and efficiency results

The speedup and efficiency using up to 72 processes are shown in Figures 5-6. The times include only the SWAN executions. Figure 5 shows very good scalability up to 24 processes when the maximum iterations were limited to five. Also shown is the speedup when unlimited iterations were used. Figure 6 shows the MPI efficiency where it can be seen that the load balancing using a maximum of five iterations significantly improves the efficiency.

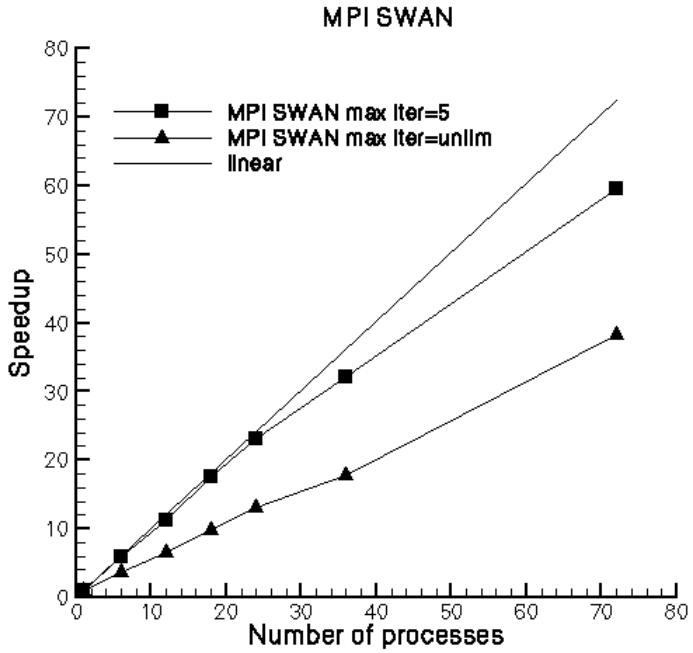

Figure 5: Speedup using 72 processes

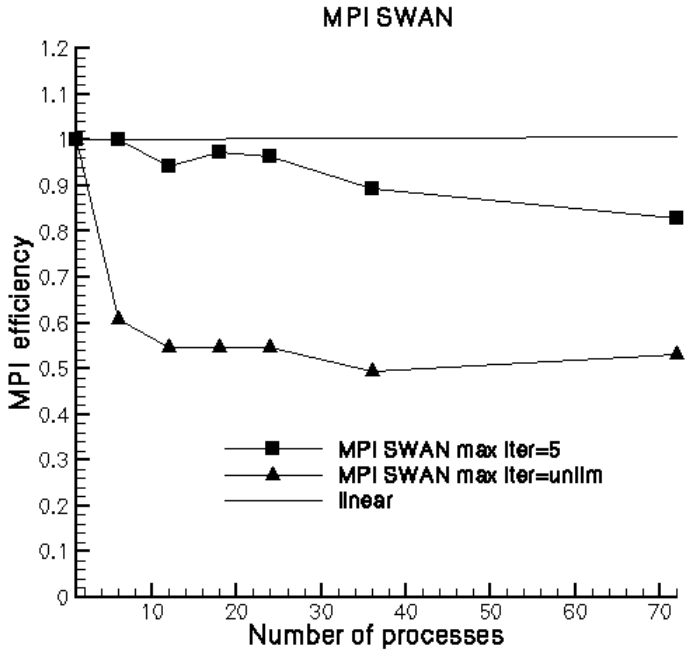

Figure 6: Efficiency using 72 processes

Figure 7 shows the time/process using unlimited iterations and a maximum of five iterations. The time/process would be constant for ideal load balancing. While Figure 7

shows improved load balancing using a maximum of five iterations, it is still not ideal and may be the reason why the efficiency falls less than 90 percent when more than 36 processes are used. Additional load balancing could be achieved by assigning data sets to a process file in such a manner that the data sets requiring the most iterations are not assigned to the same processor, which can occur when they are assigned contiguously as was done in this study.

A second plausible reason for the loss in efficiency when a large number of processes are used may be file contention in sense #2. This occurs because of the limited number of pipes to the disks where the data are located and output written. Thus as the number of processes increases, file contention in sense #2 may result as all the processes are competing for the same read and write resources. The use of MPI-IO will be explored in future studies as a means to improve the input/output performance.

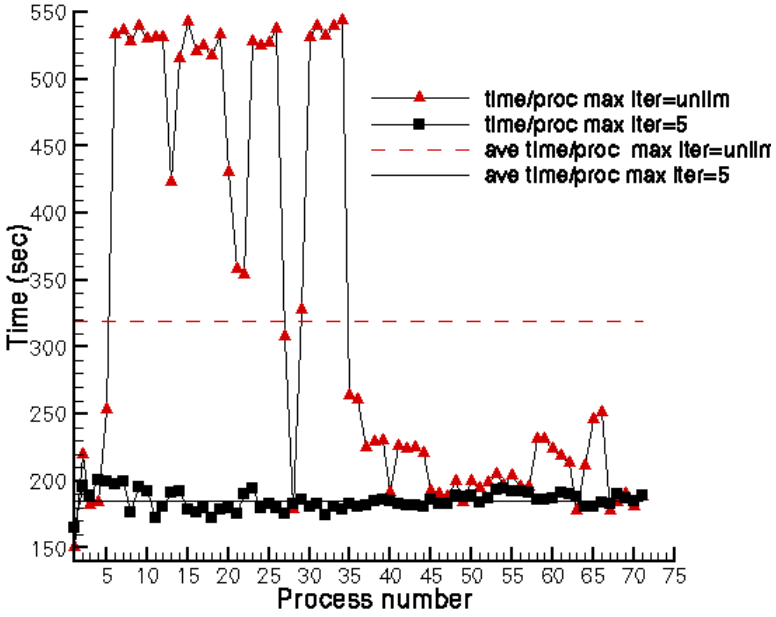

Figure 7: Time/process with max iterations = unlimited

Time-accurate vs. quasi time-accurate comparisons

The test case of Wornom et al. (2000) for a simulation of 1995 Hurricane Luis is used to compare the quasi time-accurate and the time-accurate approaches. That study employed the time-accurate version of SWAN. The sub2 nest computation from that study, which used a 96-by-96 mesh with 25 frequencies and 24 angles, is examined with both approaches. The time integration step for the time-accurate computations was 12 minutes. Both the first-order time-accurate scheme and the second-order time-accurate scheme were used with only the second order reported. The boundary spectra were created by WAM run on a larger regional mesh. Computations were made with and without depth-induced wave breaking.

This test case would not fall under assumption #1 as the domain spans approximately 131 by 131 kilometers and the wind conditions are changing quickly, particularly for the period of 9-12 September 1995. Thus, this is an interesting test case to evaluate this approach.

Figures 8-11 show comparisons between the time-accurate computation and quasi time-accurate computation at five test locations. These are NOAA buoy 44014, NOAA Station chlv2, NOAA Station dsln7, the U.S. Army Field Research Facility (FRF) buoy wr630, and the FRF 8-meter array at Duck, NC, for the 10-day period from 3-12 September 1995. NOAA buoy 44014 is the most seaward test site; the FRF 8-m array is the most landward site situated 900 meters offshore. The results for the FRF 8-m array included quadruplet wave-wave interaction (Q) and depth-induced wave breaking (B). The region is 2-by-2 degrees off the North Carolina coast at Duck.

The differences between the quasi time-accurate approach and the time-accurate approach are much smaller than the differences between the two approaches and the data. Table 1 shows the L2 norms for both approaches. The L2 norms are very close, with the quasi time-accurate approach being slightly more accurate at three of the test sites. The norms covered the period from 3-9 September the study of Wornom et al. (2001).

Table 1: L2 norms comparisons

Data site	Time-accurate	Quasi time-accurate
NOAA buoy 44014	0.634	0.586
NOAA station chlv2	0.156	0.175
NOAA station dsln7	0.554	0.573
FRF buoy 630	0.274	0.241
FRF 8-m array	0.146	0.136

Table 2 shows the wall clock times for a 3-day simulation with output requested at each hour using the time-accurate and the quasi time-accurate approach. The wall clock times for a 10-day simulation are given in Table 3. The time-accurate computation was initialized with a stationary solution at the initial output point. This additional CPU time is included in Tables 2-3.

Table 2: Wall clock time for 3-day simulation

# processors	Time scheme	Wall clock time
1	2^{nd} order time-accurate	4.58 hrs
1	1^{st} order time-accurate	3.88 hrs
1	2^{nd} order quasi time-accurate	3.34 hrs
12	2^{nd} order quasi time-accurate	17.74 min
24	2^{nd} order quasi time-accurate	8.68 min
36	2^{nd} order quasi time-accurate	6.24 min
72	2^{nd} order quasi time-accurate	3.36 min

Table 3: Wall clock time for 10-day simulation

# processors	Time scheme	Wall clock time
1	2nd order time-accurate	15.27 hrs
1	1st order time-accurate	12.93 hrs
80	2nd order quasi time-accurate	9.90 min
120	2nd order quasi time-accurate	5.72 min

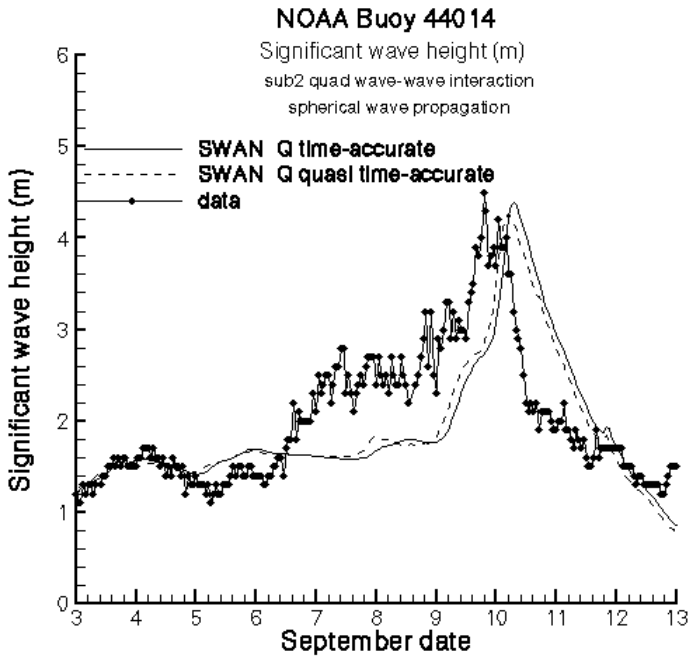

Figure 8: Time-accurate vs. quasi time-accurate: NOAA buoy 44014

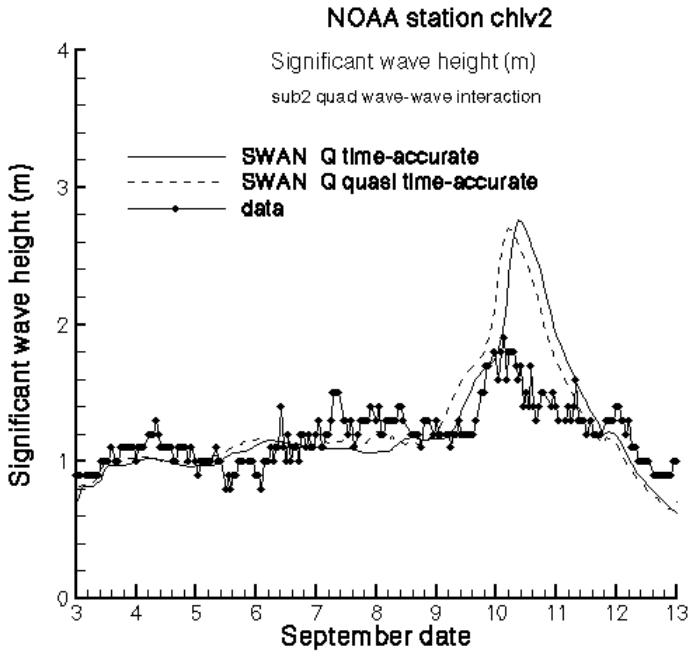

Figure 9: Time-accurate vs. quasi time-accurate: NOAA station chlv2

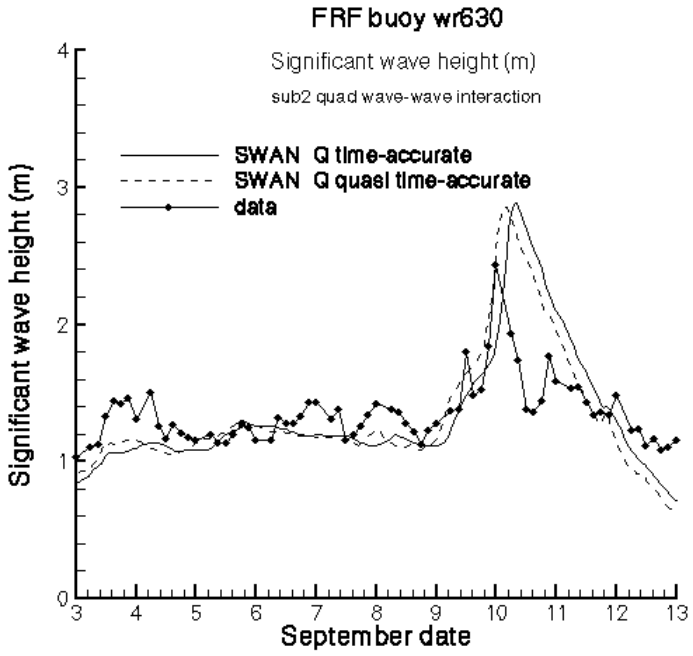

Figure 10: Time-accurate vs. quasi time-accurate: FRF buoy wr630

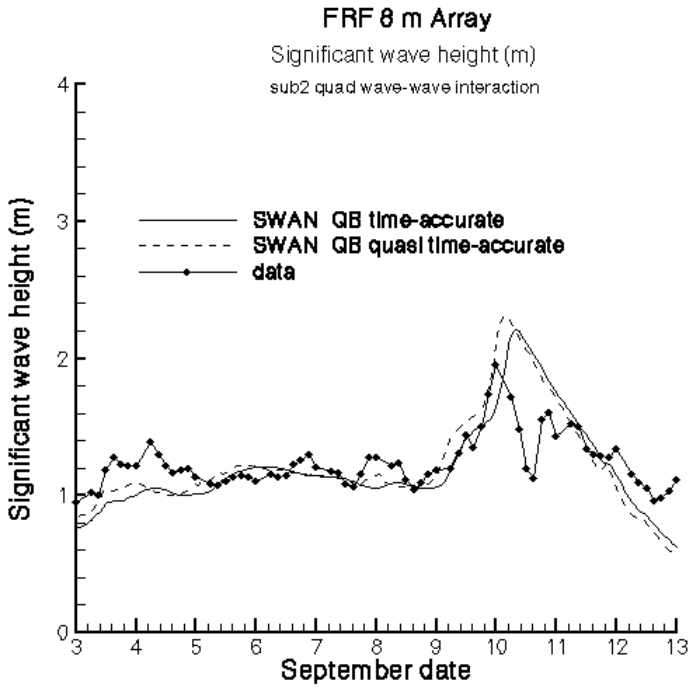

Figure 11: Time-accurate vs. quasi time-accurate: FRF 8-m array

Resources

The present implementation creates input files, named INPUT, and output files, named PRINT, on each process. A copy of the bathymetry field is also sent to each processor. The availability of processes determines the throughput time. The ERDC MSRC O3K with 512 processors is well suited for the present MPI program.

Conclusions

This study introduces a quasi time-accurate approach in which the time dependency enters the computation through the time-varying boundary spectra and wind fields; the wave action transport equation is solved as a stationary problem at each requested time. Each individual stationary integration can be regarded as disjointed so that coarse-grain parallelism may be exploited. The MPI system is used to pass the data to the parallel processes. For the 3-day, 72-hour nonstationary test case, results were obtained in 3.31 minutes using 72 processes. The same quasi time-accurate test case using one processor required 3.34 hours, thus a speedup of 59.6. The advantages of the quasi time-accurate approach are as follows:

1. Efficient MPI parallelization.
2. Equivalent accuracy with the time-accurate scheme with wall clock times reduced from hours to minutes. As such, the efficient MPI quasi time-accurate approach may be preferred to the time-accurate scheme for many DoD applications.
3. The MPI efficiency permits routine computations with the SWAN code, which are not presently practical with the time-accurate version because of long CPU times (4b-c).
4. The MPI version can be used in several modes:
    a. Faster turnaround times for the same cases
    b. Finer mesh resolution
    c. More frequencies and/or directional angles
5. The implementation of the MPI quasi time-accurate approach requires very little changes to the SWAN code itself; thus new releases of the SWAN code are easily incorporated.
6. The MPI quasi time-accurate approach was relatively easy to implement. The major changes were made over a 3-day period with a week for refinement and application to the test case.
7. The efficient MPI quasi time-accurate approach has merits and provides rapid results until a parallel time-accurate version of SWAN becomes available.

Acknowledgments

This work was supported in part by a grant of computer time from the DoD High Performance Computing Modernization Program at the ERDC MSRC in Vicksburg, MS. The author would like to thank Dr. Robert Jensen of the Coastal and Hydraulics Laboratory at ERDC for suggesting and supporting the various SWAN Programming and Environment and Training studies. The author would also like to acknowledge fruitful discussion with Dr. Richard Hanson of Rice University, Dr. Richard Weed and Dr. Nathan Prewitt of Mississippi State University, and Dr. Dan Duffy of the Computer Sciences Corporation Computational Science and Engineering group. Many discussions with Erick Rogers concerning the SWAN code are also gratefully acknowledged.

References

Booij, N., Ris, R.C., and Holthuijsen, L.H., (1999),
"A third-generation wave model for coastal regions, Part I, Model description and validation,"
J. Geoph. Research 104, C4, pp. 7649-7666.

Holthuijsen, L.H., Booij, N., Ris, R.C., Haagsma, J.G., Kieftenburg,
A.T.M.M., and Padilla-Hernandes, R., (1999),
"SWAN version 40.11 USER MANUAL."

Ris, R.C., Booij, N., and Holthuijsen, L.H., (1999),
"A third-generation wave model for coastal regions,
Part II, Verification," *J. Geoph. Research* 104 C4, pp. 7667-7681.

On Coupling the SWAN and WAM Wave Models for Accurate Nearshore Wave Predictions Stephen F. Wornom, David J.S. Welsh, Keith W. Bedford
Coastal Engineering Journal, Vol. 43 No. 3, September 2001.

STWAVE: Steady-State Spectral Wave Model Users' Manual for STWAVE, Version 3.0
Jane McKee Smith, Ann R. Sherlock, and Don Resio
U.S. Army Corps of Engineers
Engineer Research and Development Center
Coastal and Hydraulics Laboratory Report ERDC/CHL SR-01-1.